METHOD AND SYSTEM FOR CONFIGURING AND SCHEDULING SECURITY AUDITS OF A COMPUTER NETWORK

PRIORITY AND RELATED APPLICATIONS

The present application claims priority to provisional patent application entitled, "Method and System for Configuring and Scheduling Security Audits of a Computer Network," filed on January 31, 2001 and assigned U.S. Application Serial Number 60/265,519. The present application also references and incorporates herein a related U.S. non-provisional patent application entitled, "Method and System for Calculating Risk Associated with a Security Audit," filed concurrently herewith and having attorney docket number 05456.105036.

TECHNICAL FIELD

The present invention is generally directed to managing the security of a network. More specifically, the present invention facilitates the configuration and scheduling of security audits of machines in a distributed computer network.

BACKGROUND OF THE INVENTION

The security of computing networks is an increasingly important issue. With the growth of wide area networks (WANs), such as the Internet and the World Wide Web, people rely on computing networks to transfer and store an increasing amount of valuable information. This is also true of local area networks (LANs) used by companies, schools, organizations, and other enterprises. LANs are used by a bounded group of people in the organization to communicate and store electronic documents and information. LANs typically are coupled to or provide access to other local or wide area networks. Greater use and availability of computing networks produces a corresponding increase in the size and complexity of computing networks.

With the growth of networks and the importance of information available on the networks, there is also a need for better and more intelligent security. One approach to securing larger and more complex computer networks is to use a greater number and variety of security assessment devices. Security assessment devices can be used to evaluate elements in the network such as desktop computers, servers, and routers, and determine their respective vulnerability to attack from hackers. These network elements are commonly referred to as hosts

and the terms "element" and "host" are used interchangeably herein. Security assessment devices can also be used more frequently to monitor the activity or status of the elements in a computing network.

One problem with increasing the number of security assessment devices and the frequency with which they are used is deciding which elements in the network need to be audited, how frequently they should be audited, and what checks need to be run. These are decisions that often involve a variety of complicated factors and they are decisions that in practicality cannot be made every time a security audit is conducted. Increased assessment also produces a corresponding increase in the amount of security data that must be analyzed. A network administrator that is overwhelmed with security data is unable to make intelligent decisions about which security vulnerabilities should be addressed first.

An additional problem associated with maintaining adequate network security is finding the time to conduct security audits. Security audits generally must be initiated by a security professional and can hinder or entirely interrupt network performance for several hours at a time. Furthermore, existing security assessment devices typically perform a variety of security scans on a machine, some of which may not be necessary. These unnecessary scans can translate into additional "down time" for the network.

In view of the foregoing, there is a need in the art for a system which will support the auditing of a distributed computing network. Specifically, a need exists to be able to automatically survey a network and determine the role and value of each element in the network. A further need exists to be able to assess the vulnerability of each element in the network. There is also a need to automatically schedule security auditing based on the vulnerability assessment of each element and to adjust future scheduling as audit data change. In this manner, those elements deemed to have the greatest risk can be monitored more closely. Finally, a need exists to be able to manage and present data pertaining to the survey, the vulnerability assessment, and the scheduling in a convenient graphical format.

SUMMARY OF THE INVENTION

The present invention satisfies the above-described needs by providing a system and method for scheduling and performing security audits in a distributed computing environment. Assessing the security of a relatively large or complex computer network can require hundreds

of decisions about the types and timing of security checks. By facilitating the selection and scheduling of security audits, the present invention improves existing network security techniques. The present invention can identify the various elements in a distributed computing network and determine their role and relative importance. Using an element's role and relative importance, a more thorough security audit is chosen and scheduled to be run at an appropriate time. Information from the security audit can be used to calculate a security score and to modify the type and scheduling of future security audits. Security audit information can also be prioritized and presented to a user in a convenient format.

In one aspect, the present invention comprises a method for configuring and scheduling security scans of a computer network. A security audit system can conduct a discovery scan to identify elements that exist in a distributed computing network. Elements typically identified include, but are not limited to, desktop computers, servers, routers, and data storage devices. From the information collected during the discovery scan, the security audit system can determine the operating system and/or services associated with an element. The element's function and importance in the network can be used to configure an audit scan. An audit scan is a more thorough examination than a discovery scan and different types of audit scans involve different types of checks. The security audit system can schedule the selected audit scan to run at a time that will not interrupt the normal functioning of the computer network. The information collected during the audit scan can be used by the security audit system to calculate a security score for each element or group of elements. A security score is useful for identifying and prioritizing vulnerabilities that need to be remedied in the network.

In another aspect, the present invention provides a method for assessing the security of a network using a security audit system. The security audit system can receive information about elements in the network from an initial scan of the network. Using the information, the security audit system can select a more thorough audit scan to perform on a particular network. The selection of the audit scan can be based on the types of checks that need to be made on a particular element. The security audit system can also schedule the audit scan based on information collected during the initial scan. An element with greater importance or more serious vulnerabilities can be scanned more frequently than other elements in the network. Once the audit scan is performed, the security audit system receives more detailed information about

the element and a security score can be computed for the element. The security score is useful in assessing the security of the network and prioritizing issues that need to be addressed.

For yet another aspect, the present invention further provides a security audit system for configuring and scheduling security scans of a computer network. The system comprises various types of scanning engines for running different scans and an active scan engine for coordinating the selection and scheduling of the different scans. The security audit system can conduct an initial scan to assess the functions and importance of various elements in the network. The initial scan provides information for deciding when to perform a more thorough audit scan and what type of audit scan to select. A console can also be coupled to the system for communicating information concerning the scans between a user and the security audit system.

These and other aspects of the invention will be described below in connection with the drawing set and the appended specification and claim set.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating an exemplary architecture for operating an embodiment of the present invention.

FIG. 2 is a block diagram illustrating an exemplary data flow for a security audit system.

FIG. 3 is a logic flow diagram illustrating an overview of the operating steps performed by a security audit system in accordance with an exemplary embodiment of the present invention.

FIG. 4 is a logic flow diagram illustrating an exemplary process for initializing a scheduling module within a security audit system.

FIG. 5 is a logic flow diagram illustrating an exemplary process for recovering prior data within a security audit system.

FIG. 6A is a logic flow diagram illustrating an exemplary process for configuring scans with a security audit system.

FIGs. 6B, 6C, and 6D are exemplary tables associated with configuring scans.

FIG. 7 is a logic flow diagram illustrating an exemplary process for scheduling scans with a security audit system.

FIG. 8 is a logic flow diagram illustrating an exemplary process for scheduling specific scan jobs with a security audit system.

FIG. 9 is a logic flow diagram illustrating an exemplary process for scheduling an audit scan with a security audit system.

FIG. 10 is a logic flow diagram illustrating an exemplary process for running security scans with a security audit system.

FIG. 11 is a logic flow diagram illustrating an exemplary process for analyzing scan results with a security audit system.

FIG. 12 is a logic flow diagram illustrating an exemplary process for populating a host's Scan Configuration record with a security audit system.

FIG. 13 is a logic flow diagram illustrating an exemplary process for computing a host's asset value with a security audit system.

FIG. 14 is a logic flow diagram illustrating an exemplary process for computing a host's role with a security audit system.

FIG. 15 is a logic flow diagram illustrating an exemplary process for processing vulnerabilities found on a host with a security audit system.

FIG. 16 is a logic flow diagram illustrating an exemplary process for updating vulnerability state and history tables with a security audit system.

FIG. 17 is a logic flow diagram illustrating an exemplary process for processing running services found on a host with a security audit system.

FIG. 18 is a logic flow diagram illustrating an exemplary process for updating service state and history tables with a security audit system.

FIG. 19 is a logic flow diagram illustrating an exemplary process for processing a host's security score with a security audit system

FIG. 20 is a logic flow diagram illustrating an exemplary process for updating a host's security score with a security audit system

FIG. 21 is a logic flow diagram illustrating an exemplary process for shutting down a security audit system.

DETAILED DESCRIPTION OF THE EXEMPLARY EMBODIMENTS

The present invention supports the automated assessment of the security risks of a computing network. Specifically, the present invention allows a security auditing system to collect initial information about the identity and importance of elements in a computing network.

Using this initial information, the invention then provides for automatic selection and scheduling of security audit scans to be performed on the network elements. A user can provide parameters, if so desired, as to when to schedule audit scans and what types of audit scans to run. Taking the information collected from the audit scan, the auditing system can compute a security score for a network element based on its vulnerability and importance. The security score can be presented to the user in a manageable format to facilitate interpretation and response. The user may use the security score as a basis for adjusting the scheduling and configuration of future audit scans.

Although the exemplary embodiments will be generally described in the context of software modules running in a distributed computing environment, those skilled in the art will recognize that the present invention also can be implemented in conjunction with other program modules for other types of computers. In a distributed computing environment, program modules may be physically located in different local and remote memory storage devices. Execution of the program modules may occur locally in a stand-alone manner or remotely in a client/server manner. Examples of such distributed computing environments include local area networks of an office, enterprise-wide computer networks, and the global Internet.

The detailed description that follows is represented largely in terms of processes and symbolic representations of operations in a distributed computing environment by conventional computer components, including database servers, application servers, mail servers, routers, security devices, firewalls, clients, workstations, memory storage devices, display devices and input devices. Each of these conventional distributed computing components is accessible via a communications network, such as a wide area network or local area network.

The processes and operations performed by the computer include the manipulation of signals by a client or server and the maintenance of these signals within data structures resident in one or more of the local or remote memory storage devices. Such data structures impose a physical organization upon the collection of data stored within a memory storage device and represent specific electrical or magnetic elements. These symbolic representations are the means used by those skilled in the art of computer programming and computer construction to most effectively convey teachings and discoveries to others skilled in the art.

The present invention also includes a computer program that embodies the functions described herein and illustrated in the appended flow charts. However, it should be apparent that there could be many different ways of implementing the invention in computer programming,

and the invention should not be construed as limited to any one set of computer program instructions. Further, a skilled programmer would be able to write such a computer program to implement the disclosed invention based on the flow charts and associated description in the application text, for example. Therefore, disclosure of a particular set of program code instructions is not considered necessary for an adequate understanding of how to make and use the invention. The inventive functionality of the claimed computer program will be explained in more detail in the following description in conjunction with the remaining figures illustrating the program flow.

Referring now to the drawings, in which like numerals represent like elements throughout the several figures, aspects of the present invention and the preferred operating environment will be described.

Fig. 1 illustrates various aspects of an exemplary computing environment in which an embodiment of the present invention is designed to operate. Those skilled in the art will appreciate that Fig. **1** and the associated discussion are intended to provide a brief, general description of the computer network resources in a representative distributed computer environment including the inventive security audit system. The architecture comprises a console **105** and a security audit system **115** which are used to configure and schedule security audits of a network **110**. The console **105** communicates information about the current security state of the network **110** to a user. The console **105** typically comprises a graphical user interface for presenting and managing data in a convenient format for the user. The console **105** is also operable for receiving information from the security audit system **115** and allowing control of the security audit system **115**. The security audit system **115** comprises an active scan engine **120** and one or more other scan engines. In the exemplary embodiment illustrated in Fig. 1, the active scan engine **120** is coupled to an Internet scanning engine **130**, a system scanning engine **150**, and a database scanning engine **140**. Each of these scan engines illustrated in Fig. 1 is coupled to a corresponding database.

The active scan engine's **120** primary task is acquiring and maintaining current data about the configuration and security posture of the network **110**. The active scan engine **120** utilizes the subsidiary scan engines **130**, **140** and **150** as a means for gathering information about the network **110**. The network **110** typically comprises elements such as desktop computers, routers, and various servers. The active scan engine **120** is responsible for coordinating the

configuration, scheduling, and running of scans of these elements found in the network **110**. Typically, the active scan engine **120** is continuously running so that the scheduled scans can be run at their designated times, and the resultant data processed in a timely manner.

Fig. 2 illustrates a flow chart diagram of an exemplary flow of data for an embodiment of the present invention. Beginning with the user input **205**, this represents data that a user may input at the console **105** to be used by the security audit system **115** in configuring and scheduling scans. User input data **205** can include a specific network range in which to conduct scans, fixed asset and vulnerability values, and blackout periods during which security scans should not be run. The user input **205** is combined with data recovered by the discovery scan **210**. The discovery scan is an initial scan of the network **110** run by the security audit system **115**. The discovery scan is used to identify the elements on the network **110** and to assign asset values to them. An asset value is typically an arbitrary value assigned based on the importance of an element relative to other elements in a network. By identifying the function and asset value of each element on the network, the security audit system **115** can configure and schedule further scans to run on the network **110**.

The user input **205** and the discovery scan data **210** are combined to formulate state data **215** describing each of the elements in the network **110**. In step **220**, the active scan engine **120** makes decisions regarding the types of scans to be run and when they will be run on the network **110**. Ultimately, scheduled audit scans will be run against each of the elements on the network **110** in step **225**. The audit scan involves a more thorough examination of a network element than the discovery scan. The audit scan data **230** is collected and fed back into the accumulated state data **215** describing each element on the network **110**. The feedback mechanism shown in Fig. 2 enables the security audit system **115** to adjust the configuring and scheduling of future audit scans. The exemplary method illustrated in Fig. 2 allows the most important or most vulnerable elements of the network **110** to be given the highest priority in conducting security audits.

An exemplary process overview for operating the security audit system **115** is illustrated in Fig. 3. When the security audit system **115** first starts up, a scheduling module with the active scan engine **120** is initialized as shown in step **305** of Fig. 3. The initialization process involves adjusting several modules, which will be discussed in connection with Fig. 4. In step **310**, the active scan engine **120** recovers prior data from previous incompletely processed scans. The

need to recover prior data results from the active scan engine **120** being shut down when there are scan results pending processing. When a shutdown occurs, any data necessary to recover the current state of the active scan engine **120** is saved to the ASE database **125**. Upon startup, the active scan engine **120** recovers and processes any incompletely analyzed data as further illustrated in Fig 5.

In step **315**, the active scan engine **120** configures the scans that are to be run on the network **110**. The configuring of scans, discussed in greater detail in connection with Fig. 6, involves determining the type of scan to run on a network host based on its function and asset value. Once it is decided what type of scans will be run in step **315**, the active scan engine **120** chooses when to conduct scans on network elements in step **320**. The security audit system **115** can make many of the configuration and scheduling decisions that would ordinarily have to be made by the user.

In step **325**, the security audit system **115** runs the scheduled scans on the network **110**. The first time that a security audit system **115** scans the network **110** it will conduct a discovery scan to identify network elements and their function. The discovery scan collects information for use in subsequent configuring and scheduling of more thorough audit scans. After these scans are performed, the active scan engine **120** analyzes the data that are collected in step **330**. The analysis of the data can be used to readjust the configurations and scheduling of scans by returning to step **315**. Alternatively, the user can shut down the security audit system **115** in step **340**. The security audit system **115** performs tasks asynchronously from the user's perspective. When engaged in a potentially time-consuming task such as the analysis of scan results (step **330**), the active component periodically checks for a user-initiated shutdown signal. This allows the security audit system **115** to shut down in a timely manner even when engaged in lengthy tasks. As mentioned above, if a shutdown occurs when the security audit system **115** is engaged upon one or more tasks, sufficient state information is stored to allow for recovery upon reactivation. The foregoing steps are merely an exemplary embodiment of how to use the security audit system **115**. In an alternative embodiment of the invention, the foregoing steps may be performed in a different order or certain steps may be skipped entirely.

Fig. 4 illustrates an exemplary method for initializing the active scan engine **120**. In step **405**, the active scan engine **120** loads the scan blackout schedule, initializing the blackout manager **124**. The scan blackout schedule contains time periods during which scans are not to be

performed on the network **110**. The scan blackout schedule is typically determined by a user and entered using the console **105**. In step **410**, the various scanners **130**, **140**, and **150** are initialized so they will be ready to perform scans on the network **110**. The exemplary embodiment described herein discusses three types of scanning engines that can be used to perform a security audit on a network. An alternative embodiment of the present invention may employ a selected subset of these scan engines or other types of scan engines.

The Internet scanning engine **130** is a network scanning tool used to conduct the initial discovery scans performed on the network. The Internet scanning engine **130** can also be used to identify security vulnerabilities that exist across an entire network. The database scanning engine **140** performs audits of database servers identified by the Internet scanning engine **130** during the discovery scan. The system scanning engine **150** is a security auditing tool comprising software that is generally installed on individual hosts in the network. The system scanning engine **150** is typically installed on desktop computers, servers, and routers that have at least a specific asset value. Because they execute on the local host, the system scanning engine **150** is able to detect vulnerabilities that may be unidentifiable by the Internet scanning engine **130**. The active scan engine **120** works with the system scanning engine **150** to configure and schedule particular scans to be run on network elements.

In steps **415**, **420**, and **425**, components of the active scan engine **120** are initialized in preparation for conducting scans. Identified in step **415**, the decision maker component **121** receives the results of prior audit and discovery scans and determines which audit scans to run on which elements and when to run them. The job manager component **122**, initialized in step **420**, receives instructions from the decision maker component **121** and monitors what audit scans have been scheduled, when the audit scans have been scheduled, and whether the audit scans are complete. Finally, in step **425** the analyzer component **123** is initialized so that it can receive and store the results of audit scans such as a network element's functions and vulnerabilities. In alternative embodiments of the present invention, the functions performed by the analyzer component **123**, the decision maker component **121**, the blackout manager **124**, and the job manager component **122** can be performed by other components separate from the active scan engine **120**.

Fig. 5. illustrates an exemplary means for recovering prior data as referred to in step **310** of Fig. 3. The purpose of this step is to continue scanning, analysis, or decision-making work

that was interrupted during a previous shut down of the security audit system **115**. In step **505**, analysis is completed of any remaining hosts in the network **110** that have not already been analyzed. In step **510**, the active scan engine **120** checks for a user-initiated shutdown of the security audit system **115**, or proceeds to step **515** to analyze data remaining from any unprocessed scan jobs. After the prior data has been recovered, the active scan engine **120** once again checks for a user-signaled shutdown in step **520**. If a shutdown has been initiated, the active scan engine **120** shuts down; otherwise, it returns to step **315** for adjusting or configuring new scans.

An exemplary method for configuring scans is illustrated in Figs. 6A and the accompanying exemplary tables. The first time the security audit system **115** runs on a network **110**, a scan configuration table can be created after a discovery scan is performed. Subsequently, the active scan engine **120** stores the scan configuration table and, with each new scan, the table can be updated. Referring to Fig. 6A, in step **605** the active scan engine **120** determines if the decision maker **121** is due to be executed. The decision maker **121** runs periodically, or when a discovery scan finds previously unknown hosts on the network **110**. In step **610**, the decision maker **121** opens the scan configuration table in the ASE database **125**. As shown in an exemplary table in Fig 6D, the scan configuration table contains the necessary information for configuring and scheduling scans of all known hosts. In step **615**, the decision maker **121** reads a host's scan configuration record and, in step **620**, the decision maker **121** examines the host's scan configuration record. In step **625**, if the host is due for an audit scan, the host is added to the set of hosts to be scanned with the scan policy indicated in the scan configuration record. The decision maker **121** checks for a user-initiated shutdown in step **630**, and proceeds to shut down if so indicated. Otherwise, in step **635** the decision maker **121** checks for more host scan configuration records. If there are more records to read, the process **315** returns to step **615**, and is repeated with the next record. If there are no records remaining to be processed, the decision maker **121** resets itself in step **640**. This step involves setting parameters governing the next periodic run of the decision maker **121**.

The type of scan that is run on each host for an element in the network **110** can be selected manually by the user or done automatically by the active scan engine **120**. The advantage of automating the scan configuration is that there are often numerous elements to scan in a network and hundreds of possible scanning checks that can be performed on each element.

By automating the process, configuring and scheduling scans of each element of the network can be performed periodically, or whenever a new element is found during a discovery scan Fig. 7 illustrates an exemplary decision step for determining whether to schedule a discovery scan or an audit scan. A discovery scan is conducted periodically, or when a network is being audited for the first time. If a discovery scan is being conducted in step **710**, the active scan engine **120** can follow the exemplary method for scheduling a job illustrated in Fig. 8. If an audit scan is going to be conducted in step **715**, the active scan engine **120** locates other hosts with the same scan policy as shown in Fig. 9.

A scan policy comprises a list of vulnerabilities to be checked during a scan. Scanning all the hosts with the same policy at one time allows the active scan engine **120** to coordinate scans efficiently. Once the other hosts with the same policy are located in step **905**, a job is scheduled for the audit scan in step **910** in the same way that a job is scheduled for a discovery scan. If more host ranges have been configured for scans, step **915** will return to step **905** and the process will repeat for the next host range. Otherwise, the process is complete.

An exemplary method for scheduling a job is illustrated in Fig. 8. In step **805**, a unique job identifier is assigned to the scan job. In step **810**, the active scan engine **120** estimates the duration of the scan and checks with the blackout manager, in step **815**, for a clear time period of at sufficient duration for scanning. Once a time period is selected for the scan job, in step **820** a start time is scheduled. In step **825**, the job is registered with the job manager **122** in the active scan engine **120** so that it can be tracked. In step **830**, if the scheduled job is for a discovery scan, the process is complete. Otherwise, in step **835** the last scan job identifier is updated for each host included in the newly scheduled job.

Once a job is scheduled, the scan is ready to run against the appropriate elements in the network **110**. An exemplary method for running a scan, as referred to in step **325** of Fig. 3, is illustrated in greater detail in Fig. 10. In step **1005**, the active scan engine **120** retrieves the next available job from the job manager. Depending on the type of scan that is to be run, in step **1010**, the active scan engine **120** starts the appropriate scan engine. In steps **1015** and **1020**, the active scan engine **120** sends a request to the appropriate scan engine to start the job and updates the job status to active. Running the scan collects data about one or more of the various elements in the network and returns that data to the active scan engine **120** for analysis.

Referring to Fig. 11, an exemplary method for analyzing scan results, as referenced in step **330** of Fig. 3 and step 515 of Fig. 5, is illustrated. In step **1105**, the analyzer component **123** selects the next scan job to analyze. In step **1110**, the analyzer **123** selects the next host to process in the current scan job's data. If this is the first time that the host is examined, then the data are from a discovery scan. The current host's scan configuration record is written or updated in step **1115**. Based on the host function and asset value, further scan policies can also be automatically selected by the scan engine for future audit scans against the host. In steps **1120** and **1125**, the analyzer **123** processes the vulnerability and service records for the current host.

If the scan was a discovery scan, the analysis process ends at step **1130**. However, if the scan was an audit scan, the analysis process continues in step **1135,** where a new security score is computed for the host. An advantageous means for calculating a security score is described in U.S. non-provisional patent application entitled "Method and System for Calculating Risk Associated with a Security Audit," filed concurrently herewith, having attorney docket number 05456.105036. An exemplary method for processing a security score is discussed in greater detail with reference to Fig. 19. In step **1140**, the host's previous security score is updated. If the user initiated a shutdown of the security audit system **115** at this time, any unanalyzed host information or job information is saved in steps **1150** and **1155**. If there are more hosts to be analyzed or more scan jobs to be performed in steps **1165** and **1170**, the process will return to the beginning and repeat.

In Fig. 12, an exemplary method for populating a host's scan configuration, as referenced in step **1115** of Fig. 11, is illustrated. In step **1203**, the analyzer **123** examines the state data for the current host. These data indicate what operating system and services a host is running, and form the basis for computing the host's asset value in step **1205** and its role in step **1210**. In step **1215**, the analyzer **123** retrieves the scan policy to be applied to the host based on its asset value and role. Fig. 6B is an exemplary table that maps a host role and asset value to a scan policy. In step **1218**, the scan frequency of the host is computed as a function of its asset value. Fig. 6C illustrates an exemplary table for mapping a host's asset value to a scan frequency. Both of the mapping tables illustrated in Figs. 6B and 6C may be adjusted and customized by the user. The analyzer **123** writes or updates the host's scan configuration record in step **1223**.

Fig. 13 illustrates an exemplary method for computing a host's asset value, as referenced in step **1205** of Fig. 12. In step **1305**, the analyzer **123** retrieves from the ASE database **125** the asset value associated with the host's operating system. Asset values assigned on the basis of operating system reflect the greater importance of servers and routers than regular desktop systems. In step **1310**, the analyzer **123** retrieves the asset value associated with the host's running services. Asset values are associated with a number of services in the ASE database **125**, and reflect the relative importance of the various services. In step **1310**, the analyzer **123** selects the maximum asset value associated with any of the services that are active on the host. In step **1315**, the analyzer **123** retrieves the asset value associated with the host's vulnerabilities. As with services, the presence of some vulnerabilities can indicate a greater importance for a host; the asset values associated with vulnerabilities in the ASE database **125** reflect this. In the present example, the maximum asset value associated with any of the host's vulnerabilities is chosen. In step **1320**, the analyzer **123** selects the maximum of the previously retrieved asset values as the host's asset value.

Fig. 14 illustrates an exemplary method for computing a host's role, as referenced in step **1210** of Fig. 12. Steps **1405**, **1425**, and **1445** distinguish between various exemplary host operating systems. The next layer of decisions (steps **1410** and **1430**) distinguish between web servers and non-web servers. As shown in the figure, the exemplary host roles identified are NTDesktop (step **1415**), NTWebServer (step **1420**), UnixWebServer (step **1435**), UnixDesktop (step **1440**), Router (step **1450**), and Unknown (step **1455**). While Fig. 14 shows an exemplary implementation of determining a host's role on a network, those skilled in the art will recognize that other host parameters may be taken into account, yielding a greater variety of roles.

Referring to Fig. 15, an exemplary method for processing vulnerability records is illustrated. In step **1505**, the analyzer **123** queries the vulnerability history table for the current host. In step **1510**, a vulnerability identified during the audit scan is located and the vulnerability history table is examined for the same listing in step **1515**. If the vulnerability is listed in the history table, the state table and history table are updated for the particular host in step **1520**. As shown in greater detail in the exemplary logic flow diagram in Fig. 16, step **1520** involves the analyzer **123** updating the record in the state table and the record in the history table. If the vulnerability is not found in the history table in step **1515**, the analyzer **123** writes a new record to the state table and history table in steps **1530** and **1535**. If there are more vulnerabilities to be

examined, the process returns to step **1510**. Once the vulnerability records for a host are processed, the logic flow diagram returns to step **1125** for processing service records.

An exemplary method for processing service records, as referred to in step **1125** of Fig. 11, is illustrated in Fig. 17. In step **1705**, the analyzer **123** queries the service history table for the current host that is being examined. In step **1710**, a service identified during the audit scan is located and the service history table is examined for the same listing in step **1715**. If the service is listed in the history table, the analyzer **123** updates the state table and history table for the particular host in step **1720**. As shown in greater detail in the exemplary logic flow diagram in Fig. 18, step **1720** involves updating the record in the state table and the record in the history table. If the service is not found in the history table in step **1715**, the analyzer **123** writes a new record to the state table and history table in steps **1730** and **1735**. If there are more services to be examined, the process returns to step **1710**. Once the service records for a host are processed, the logic flow diagram returns to step **1130**.

An exemplary method for processing the security score of a host is illustrated in Fig. 19. In step **1905**, the active scan engine **120** queries the vulnerability state table for the set of vulnerabilities for the current host. The vulnerabilities were previously detected by the various scans performed on the particular host. The active scan engine **120** selects the next vulnerability in the state table in step **1910** and calculates a risk for the vulnerability in step **1915**. An exemplary method for calculating a risk in association with a security audit of a computer network is taught in the related application referenced herein. In step **1920**, the vulnerability count for the risk band that includes the calculated risk is incremented. If there are additional vulnerabilities in the state table, the process is repeated and the vulnerability count for the appropriate band is incremented. When there are no remaining active vulnerabilities for this host in the state table, a logarithmic band calculation is applied to the accumulated risks in step **1930**. An exemplary method for performing a logarithmic band calculation on accumulated risks is taught in the related application referenced herein.

The security audit system **115** keeps a record of security scores over time. In step **2005** of Fig. 20, the most current host security score is retrieved from the host table. If the most current security score is different from the newly calculated score in step **2010**, the active scan engine **120** writes a new record to the host history table in step **2015**, and updates the host's current security score in the host table in step **2020**. If the security score is not different in step

2010, the process returns to step **1145** in Fig. 11. The forgoing steps illustrate an exemplary method for processing the security score of the host. In alternative embodiments of the invention other methods can be used to compute security scores for various elements in a distributed computing network.

Fig. 21 illustrates an exemplary method for shutting down the active scan engine **120**, as referenced in various figures. The shutdown procedure consists of saving any state information necessary to resume any operations in progress at the time of shutdown. This information consists of information about jobs scheduled but not completed, jobs completed but not analyzed, and jobs incompletely analyzed.

In conclusion, the present invention enables and supports security auditing of a distributed computing network. The security audit system can conduct a discovery scan of the network to identify network elements and determine their function, vulnerabilities, and relative importance. Using this information, more comprehensive audit scans are scheduled to regularly assess and monitor the security of the network. The security audit system can automatically select particular audit scans based on the types of hosts identified in the network. The audit scans can be automatically scheduled so as not to interfere with the regular functions of the network. Information collected during the audit scans can also be used to compute a security score for a network element.

It will be appreciated that the present invention fulfills the needs of the prior art described herein and meets the above-stated objects. While there has been shown and described the preferred embodiment of the invention, it will be evident to those skilled in the art that various modifications and changes may be made thereto without departing from the spirit and the scope of the invention as set forth in the appended claims and equivalence thereof. Although the present invention has been described as operating on a local area network, it should be understood that the invention can be applied to other types of distributed computing environments. Furthermore, it should be readily apparent that the components of the security audit system can be located in various local and remote locations of a distributed computing environment.